Algorithms

Activity 1: Learner Sheet A

You have just sorted your items using your own methods.

Now sort the items based on **COLOUR**.

You will have to write your very first algorithm for this to document your solution.

Either complete this task using pseudo code or flow charting.

Pseudocode

Pick up Item

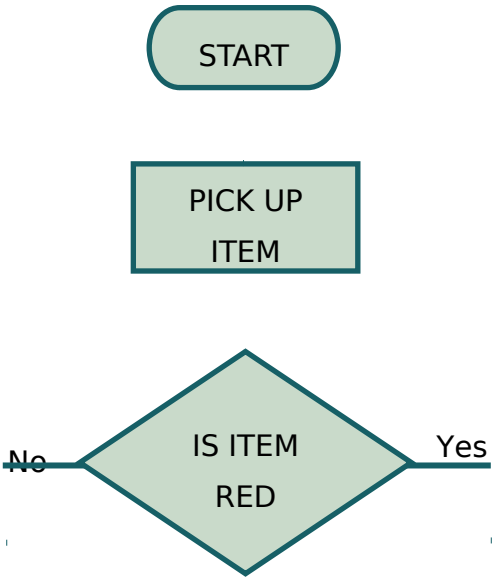
If Item is **RED**

Place at front

ELIF Item is **BLUE**

Place at Back

Flowchart



Activity 1: Learner Sheet B

You have just sorted your items using your own methods.

Now sort the items based on **SIZE**.

You will have to write your very first algorithm for this to document your solution.

Either complete this task using pseudo code or flowcharting.

Pseudocode

Pick up Item

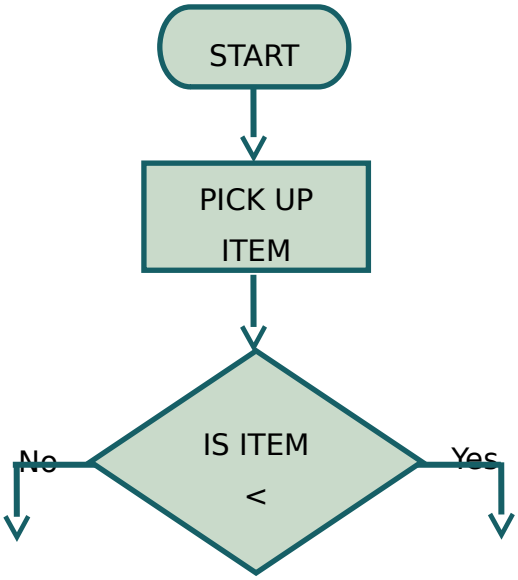
If Item is smaller than

Place in pile

ELIF Item is Bigger than

Skip

Flowchart



OCR
Oxford Cambridge and RSA

Activity 1: Learner Sheet C

You have just sorted your items using your own methods.

Now sort the items based on **TYPE.**

You will have to write your very first algorithm for this to document your solution.

Either complete this task using pseudocode or flow charting.

Pseudocode

Pick up Item

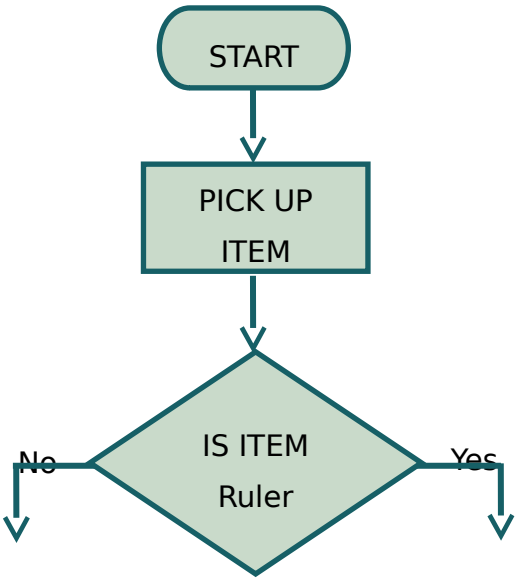
If Item is a Ruler

Place at front

ELIF Item is Rubber

Place at Back

Flowchart



OCR
Oxford Cambridge and RSA

Activity 2

Task 1 Sorting Algorithms

You have been provided with a sorting algorithm. Use the task sheet below to complete the questions as you test out the programming code.

You will need to understand in detail how this algorithm works.

1. `List = [43,21,12,80,3,2,35]`

What is this used for?

2. `End = len(List)-1`

What is this used for?

3. `While (end != -1) :`

What is this used for?

```
4. For i in range(0, end):
       if List[i] > List[i+1]:
        temp = List[i]
        List[i] = List[i+1]
        List[i+1]= temp
```

What is this used for?

In your pairs discuss the sorting algorithm you are currently using and the comments you have provided about the code.

Make notes below as you will need to discuss your view in the group.

Task 2 Investigating Sorting Algorithms

Using the sorting algorithm you have been working on within your pair test out the provided lists. Document your findings from sorting the different lists using your algorithms.

List 1: [23,67,89,12,3456,12,34,567,12]

List 2: [1,2,3,4,5,6,7,8,9,10]

List 3: [10,2,3,4,5,6,7,8,9,1]

List 4: [1,2,3,4,5,45,67,23,34,2]

List 5: [456,24,56,12,26]

1. Which list was the fastest list to **SORT** and why?

2. Which list was the slowest list to **SORT** and why?

3. List the weaknesses you feel this algorithm has.

Activity 3

Task 1

Watch the video provided to learn about the concept of complexity. This is an introduction video to slowly get you into the topic.

1. What is Big O notation?

2. What is Big O notation a measure of?

3. Using what you have seen on the video what part of this calculation has the greatest impact on the answer?

   20n^2 + 23 + 45n^4 =

4. Explain the concept of O(1) in your own words.

Task 2 Big O notation

There are rules with the big O notation that you need to understand and be able to explain. This worksheet will provide you with an opportunity to test your understanding of this.

O(1) is the no-growth curve. An O(1) algorithm's performance is conceptually independent of the size of the data set on which it operates.

O(N) says that the algorithm's performance is directly proportional to the size of the data set being processed. Scanning an array or linked list takes O(N) time.

O(N^2) says that the algorithm's performance is proportional to the square of the data set size. This happens when the algorithm processes each element of a set, and that processing requires another pass through the set. The infamous Bubble Sort is O(N^2).

Using the comments on the previous page try and label the 4 snippets of code below with the correct Big O notation. You will have to discuss your answer.

1.
```
while (end != -1):
        swapped = -1
        for i in range(0, end):
             if List[i] > List[i]
              temp = List[i]
              List[i] = List[i+1]
              List[i+1]= temp

              print(List)

              swapped = i
  end = swapped
```

2.
```
for i in range(len(CallList)):
        Call_Queue.put(CallList[i])
```

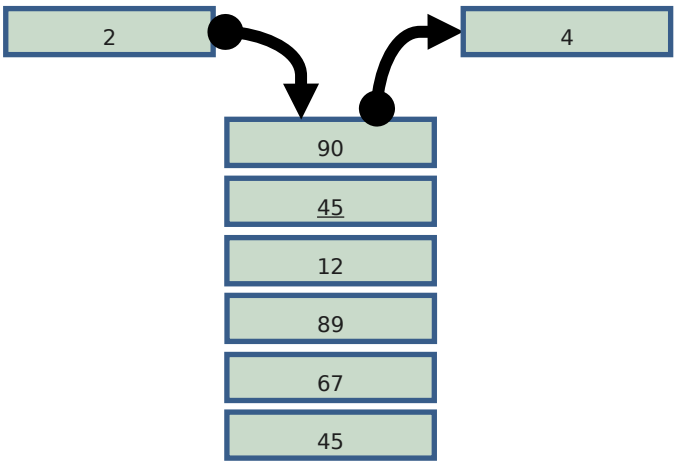
3.
```
def Port(x,y,z):
        minecraft.player.setPos(x,y,z)
```

Activity 4: Learner Sheet

Task 1

Below are two images. Using the code provided to you, state what data structure each is and explain in your own words how the data structure works.

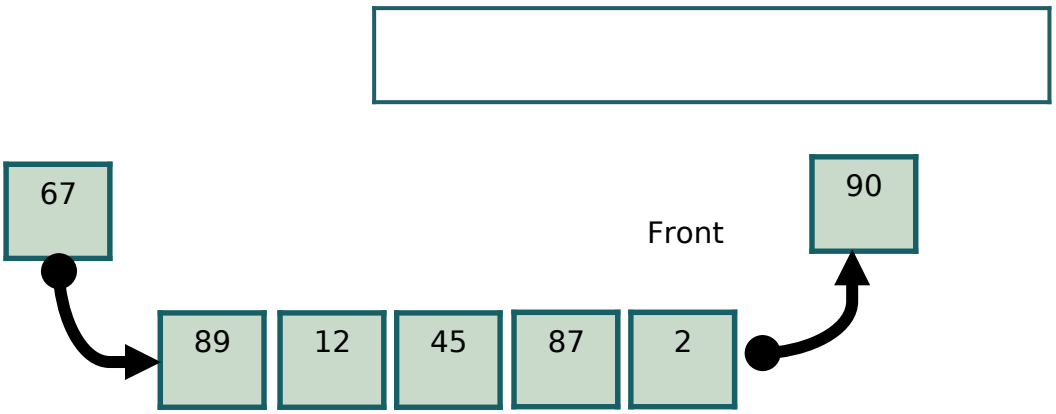
1. Name of Data Structure



How the data structure works:

OCR
Oxford Cambridge and RSA

2. Name of Data Structure:



How the data structure works:

Task 2

Using the code provided to you in the previous task as a starting point produce the code for one of the real life problems in pairs using one of the two data types.

Your teacher will show you an example on the board of a working example.

Produce a programmed solution for one of the following problems:

- Game login
- Call centre
- Robot plate clearer at restaurant
- Robot librarian putting books away.

When programming in pairs you will take turns programming and swap every 5 minutes. Complete the programming log for your partner as they program.

Pair Programming Log

Name of Driver:

What did they do?

Extra Support:

In this part you discuss what the driver is currently programming. What code are they developing?

Errors or suggested improvements:

Extension Work:

Remember the reason you are coding in this manner is to learn from each other.

Make sure you are going into detail and cover as many points as you can to produce a much higher set of feedback.

Think about improvements

Areas for development: To be completed by the driver later:

Important points to remember:

The aim of programming in this manner is to support your own development. Make sure that you complete this sheet well to get the most out of the task